Triakis Corporation

Executable Specification Implementation Document

For the

Shuttle Remote Manipulator System

A NASA CI03
SARP Initiative 583
IVV-70 Project

Table of Contents

Table of Figures

Table of Tables

1 Introduction

This specification is being developed to support a research project funded by the NASA Software Assurance Research Program (SARP) during the fiscal year 2003 Center Initiatives (CI03) effort. A system-level, executable specification (ES) based simulation of the Shuttle Remote Manipulator System (SRMS) has been created from the requirements specified in the System Requirements (SARP-I583-001) and Simulator Requirements (SARP-I583-002) Specifications, and will be used as a vehicle for exploring the concepts described in section 2 of Triakis proposal number TC_G020614.

The format and content of this specification is designed to follow the System Requirements (SARP-I583-001) and Simulator Requirements (SARP-I583-002) Specifications from which the virtual system has been developed. As our project effort progresses, this specification will be updated to reflect changes to the scope and fidelity of system requirements due to an improved understanding of the extent that our virtual SRMS must be developed to support our research goals.

This document describes how the system design described in the System Design Document (SARP-I583-101) has been implemented as an ES-based simulator. This simulator will be used to evaluate the extent to which the Triakis concept of Executable Specifications (ES') achieves unambiguous communication of system requirements thereby reducing errors induced by interpretation of ambiguous specifications. It will also be used to evaluate the potential that substituting a detailed executable (DE) hardware simulation running actual embedded software, in place of the ES, has for reducing costs and maintaining test consistency through reuse of unmodified system level tests.

Further, new methods of gathering software metrics through use of the simulator will be sought, explored, and evaluated. The virtual system simulator developed for this project will be used to evaluate other potential benefits that its virtual system integration laboratory (VSIL) environment offers in support of general testability, independent validation & verification (IV&V), reliability, and safety.

1.1 System purpose

The system specified herein is intended to represent the SRMS in a general sense only. The system requirements laid out in this document will form the basis for creating a virtual system simulator that will be used as a vehicle to facilitate the research goals stated in Triakis proposal number TC_G020614. As such, system components and functions of the real-world SRMS that are not required to support our research goals have been omitted.

While the purpose of the actual SRMS is to facilitate the deployment and retrieval of shuttle payloads as well as extra-vehicular activity missions, the derivative SRMS will not incorporate functioning end-effectors required for these purposes. The specified SRMS will demonstrate the control and movement capability of the RMA along with integral shuttle orbiter-protective safeguards.

1.2 Scope

The SRMS simulator implementation partially documented herein is a system-level, ES simulation that approximately models a subset of the system characteristics of the existing NASA space shuttle RMS. Adaptations to the functionality of the actual SRMS have been incorporated to the extent required for the stated research purposes and demonstration of the research results.

This document specifically addresses key implementation details of the RMS Computer ES, the only part within the SRMS simulator that will be developed into a Detailed Executable (DE). The RMS Computer ES has been developed to the requirements specified in the System Requirements Specification (SARP-I583-001), the Simulator Requirements Specification (SARP-I583-002), and the System Design Document (SARP-I583-101).

1.3 Definitions, acronyms, and abbreviations

AFDX	Avionics Full Duplex Switched Ethernet
CCTV	Closed-Circuit Television
CI03	Center Initiative for fiscal year 2003
C/W	Caution/Warning
DE	Detailed Executable
ES	Executable Specification
EVA	Extra Vehicular Activity
IV&V	Independent Verification and Validation
N/A	Not Applicable
NASA	National Aeronautics & Space Administration
OSMA	Office of Safety and Mission Assurance
PDRS	Payload Deployment and Retrieval System
RHC	Rotational Hand Controller
RMA	Remote Manipulator Arm
RMS	Remote Manipulator System
RMSC	RMS Computer
RMSCP	RMS Control Panel
SARP	Software Assurance Research Program
SimRS	Simulator Requirements Specification
SRMS	Shuttle Remote Manipulator System
SyDD	System Design Document
SyRS	System Requirements Specification
THC	Translational Hand Controller
VSIL	Virtual System Integration Laboratory

1.4 References

http://spaceflight.nasa.gov/shuttle/reference/index.html NASA Shuttle Reference web site
http://science.ksc.nasa.gov/shuttle/technology/sts-newsref/sts-deploy NASA PDRS web page
ISBN 0-345-34181-3 Joels, Kennedy & Larkin; Ballantine books, 1988:
The Space Shuttle Operator's Manual (Revised Edition)
SARP-I583-001 System Requirements Specification for the Shuttle Remote Manipulator System
SARP-I583-002 Simulator Requirements Specification for the Shuttle Remote Manipulator System
SARP-I583-101 System Design Document for the Shuttle Remote Manipulator System
SARP-I583-204 Ancillary Simulator Parts Document for the Shuttle Remote Manipulator System
SARP-I583-205 System Test Design Document for the Shuttle Remote Manipulator System
TC_G020614 Triakis proposal to NASA for the SARP (Solicitation No: NRA SARP 0201), 14 June 2002

1.5 SRMS overview

While this specification addresses requirements for the simulator itself, those requirements are, to a great extent driven by the system requirements. Consequently, the following excerpt from the NASA PDRS web page has been included here to provide an overview of the real-world SRMS:

> *The payload deployment and retrieval system (PDRS) includes the electromechanical arm that maneuvers a payload from the payload bay of the space shuttle orbiter to its deployment position and then releases it. It can also grapple a free-flying payload, maneuver it to the payload bay of the orbiter and berth it in the orbiter. This arm is referred to as the remote manipulator system (RMS).*
>
> *The shuttle RMS is installed in the payload bay of the orbiter for those missions requiring it. Some payloads carried aboard the orbiter for deployment do not require the RMS.*

The RMS is capable of deploying or retrieving payloads weighing up to 65,000 pounds. The RMS can also be used to retrieve, repair and deploy satellites; to provide a mobile extension ladder for extravehicular activity crew members for work stations or foot restraints; and to be used as an inspection aid to allow the flight crew members to view the orbiter's or payload's surfaces through a television camera on the RMS.

2 General system description

The simulator designer used the following excerpt from the NASA PDRS web page as a reference source and it is given here to provide a general SRMS description for informational purposes only.

The basic RMS configuration consists of a manipulator arm; an RMS display and control panel, including rotational and translational hand controllers at the orbiter aft flight deck flight crew station; and a manipulator controller interface unit that interfaces with the orbiter computer. Normally, only one RMS is installed during a shuttle mission, on the left longeron of the orbiter payload bay.

The RMS arm is 50 feet 3 inches long, 15 inches in diameter, and has six degrees of freedom. The six joints of the RMS correspond roughly to the joints of the human arm with shoulder yaw and pitch joints; an elbow pitch joint; and wrist pitch, yaw and roll joints. The end effector is the unit at the end of the wrist that actually grabs, or grapples, the payload.

The RMS can only be operated in a zero gravity environment, since the arm dc motors are unable to move the arm's weight under the influence of Earth's gravity. Each of the six joints has an extensive range of motion, allowing the arm to reach across the payload bay, over the crew compartment or to areas on the undersurface of the orbiter. Arm joint travel limits are annunciated to the flight crew arm operator before the actual mechanical hard stop for a joint is reached.

One flight-crew member operates the RMS from the aft flight deck control station, and a second flight-crew member usually assists with television camera operations. This allows the RMS operator to view RMS operations through the aft flight deck payload and overhead windows and through the closed-circuit television monitors at the aft flight deck station.

The orbiter's CCTV aids the flight crew in monitoring PDRS operations. The arm has provisions on the wrist joint for a CCTV camera that can be zoomed, a viewing light on the wrist joint and a CCTV with pan and tilt capability on the elbow of the arm. In addition, four CCTV cameras in the payload bay can be panned, tilted and zoomed. Keel cameras may be provided, depending on the mission payload. The two CCTV monitors at the aft flight deck station can each display any two of the CCTV camera views simultaneously with split screen capability. This shows two views on the same monitor, which allows crew members to work with four different views at once. Crewmembers can also view payload operations through the aft flight station overhead and aft (payload) viewing windows.

The arm has a number of operating modes. Some of these modes are computer-assisted, moving the joints simultaneously as required to put the end point (the point of resolution, such as the tip of the end effector) in the desired location. Other modes move one joint at a time; e.g., single mode uses software assistance and direct and backup hard-wired command paths that bypass the computers.

Four RMS manually augmented modes are used to grapple a payload and maneuver it into or out of the orbiter payload retention fittings. The four manually augmented modes require the RMS operator to use the RMS translational hand controller (THC) and rotational hand controller (RHC) with the computer to augment operations.

The THC and RHC located at the aft flight deck station are used exclusively for RMS operations. The THC is located between the two aft viewing windows. The RHC is located on the left side of the aft flight station below the CCTV monitors. The THC and RHC have only one output channel per axis. Both RMS hand controllers are proportional, which means that the command supplied is linearly proportional to the deflection of the controller.

There are two types of automatic modes that can be used to position the RMS arm: operator-commanded and preprogrammed.

The operator-commanded automatic mode moves the end effector from its present position and orientation to a new one defined by the operator via the keyboard and RMS CRT display. The arm moves in a straight line to the desired position and orientation and then enters the hold mode.

The preprogrammed auto sequences operate in a manner similar to the operator-commanded sequences. Instead of the RMS operator entering the data on the computer via the keyboard and CRT display, the RMS arm is maneuvered according to a command set programmed before the flight, called sequences. Each sequence is an ordered set of points to which the arm will move. Up to 200 points may be preprogrammed into as many as 20 sequences.

The description provided is intended to give a general picture of system functionality upon which our virtual system has been modeled. The features actually implemented and the fidelity of this virtual SRMS representation are dependent upon what is needed to support our overall research goals.

Most of the parts comprising the simulated SRMS have been implemented as abstract, high-level ES parts, while other ES parts have been simulated with great fidelity. The SRMS control computer (RMSCC) part has been implemented as a high-fidelity ES around which the system-level test cases are being developed.

Unless otherwise indicated, subsequent references to all elements of the SRMS and surrounding systems within this document are to be construed as referring to the virtual system elements within the simulator being developed and not the actual SRMS (in use on the NASA shuttle program) on which the virtual system is based.

3 Simulator performance characteristics

The SRMS simulator was created using the Triakis IcoSim simulator application running in a Microsoft Windows® XP operating system environment. [SimRS: R1]

The SRMS simulator makes use of standard computer and mouse devices for user interaction. [SimRS: R2]

The virtual SRMS has implemented all of the requirements identified in the released version of the SyRS for the SRMS (Triakis document SARP-I583-001). [SimRS: R3]

3.1 System context

The SRMS described in the SyRS is intended be a self-contained system with few connections to the virtual shuttle within which it will function. Figure 1 shows a screenshot of the simulated RMA within the virtual shuttle orbiter. Neither the manipulator positioning mechanism nor a functioning end effector has been implemented in this SRMS.

The SRMS is rendered as a 3-D graphical representation of the shuttle orbiter cargo bay with the RMA attached to the portside cargo door support longeron as indicated in Figure 1. [SyRS: R1, SimRS: R4, SimRS: R5]

The SRMS draws its power from the space shuttle 28VDC and 115VAC/400Hz power supplies as required to function as described herein. [SyRS: R2, SimRS: R6]

The RMS control & display panel and the closed circuit television (CCTV) monitors that the crew employs in the operation of the SRMS are located on the orbiter flight deck at the aft crew station. [SyRS: R3, SimRS: R7]

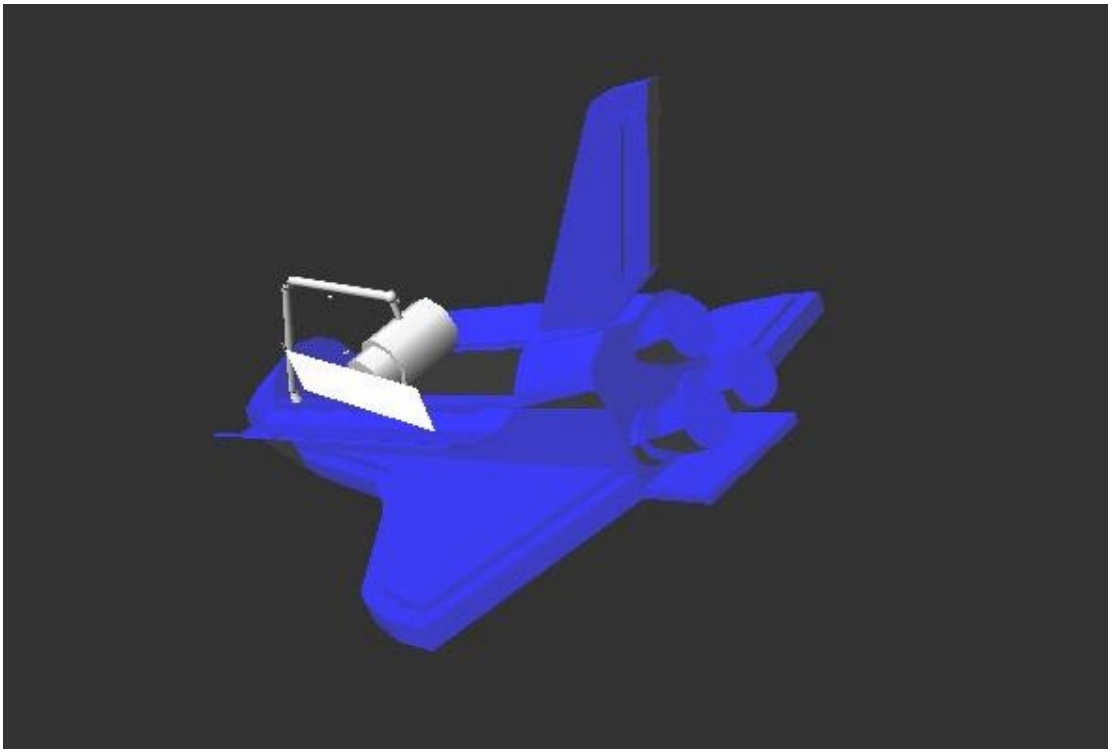


Figure 1: Simulated RMA Within Shuttle Orbiter

3.2 Major system components

The SRMS simulator comprises three principal simulated elements: [SyRS: R4, SimRS: R8]

a) A remote manipulator arm (RMA) (Figure 1),
b) A RMS control & display panel (Figure 2), and
c) A RMS control computer (RMSCC).

CCTV monitors have been simulated for visually monitoring RMA activity during operation. [SyRS: R5, SimRS: R9]

The RMSCC provides the interface between the RMS control & display panel and the RMA itself. [SyRS: R6, SimRS: R10]

All elements within the virtual system have been implemented as ES' with a level of detail commensurate with the element's importance to our research interests. [SimRS: R11]

The RMSCC has been implemented as a high-fidelity ES. [SimRS: R12]

The RMSCC ES has been validated and verified through the application of a comprehensive suite of tests designed to exercise and verify conformance to the functional system requirements. [SimRS: R13]

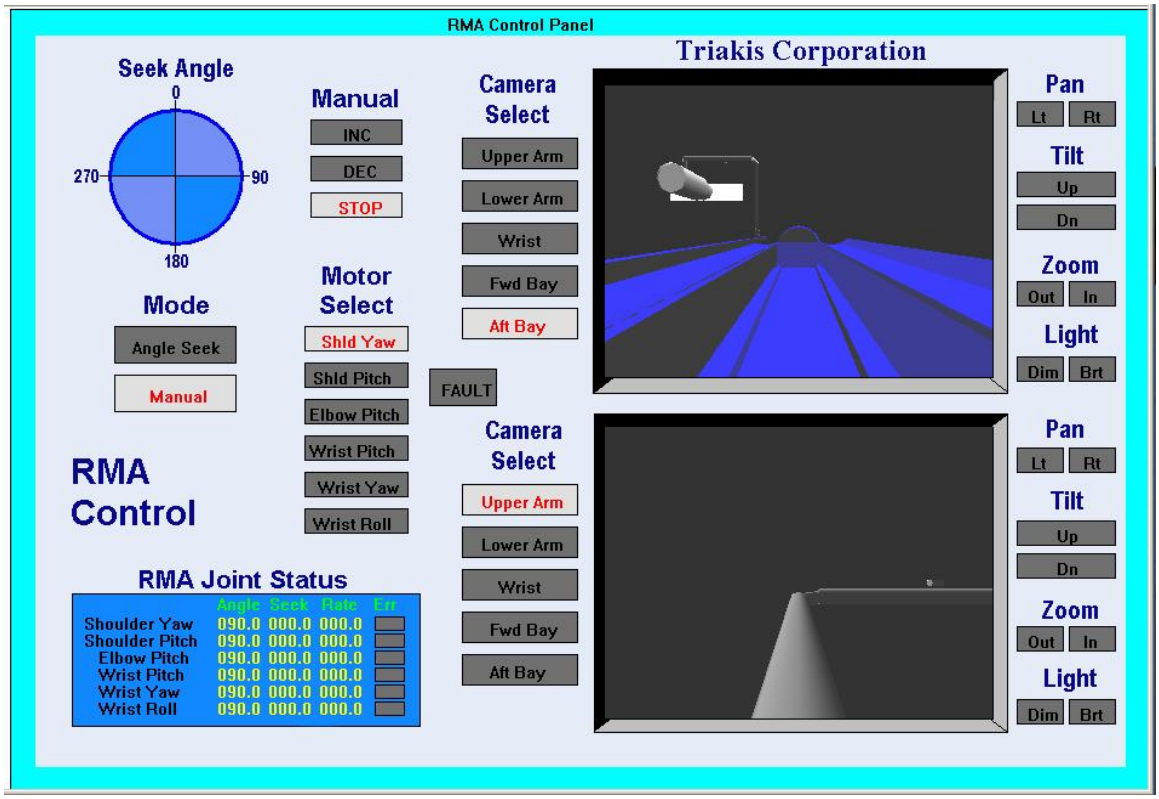


Figure 2: Simulator RMS Control & Display Panel

3.3 Simulator implementation

3.3.1 Overview

The SRMS simulator implements the design described in the SDD as a hierarchical collection of interconnected parts that represent hypothetical or actual real world components. Each high-level part contains successively lower-level parts and so on until the desired level of detail has been reached. Parts that contain other parts but provide no additional functionality beyond that of the parts it contains is referred to as a container part. Most of the functionality of a part is implemented at the lower levels of its part tree and as a result, many of the high-level parts are simply container parts.

Part functionality may be implemented as a model of a specific part such as a resistor, A/D converter, or microcontroller. Alternatively, part functionality may be simulated as a model of a group of real-world parts without considering each part independently. A power supply, for example, may be modeled according to its specifications (i.e. voltages, ripple, current limits, tolerances, power dissipation, efficiency, etc.) or a complex analog or digital circuit may be modeled according to the equations that define it.

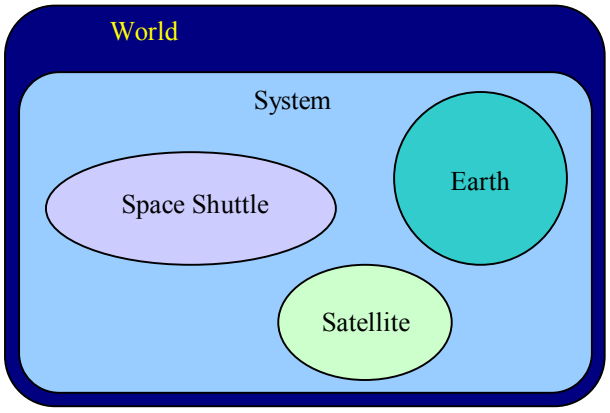


Figure 3: High-level SRMS Simulator Object Hierarchy

The diagram shown in Figure 3 depicts the high-level object hierarchy of the SRMS simulator. The "World" of this simulation contains a "System" within which the "Earth," "Space Shuttle" & "Satellite" objects operate and interact. The "World" also contains a hidden object, called the "Background," that is used to facilitate automatic testing.

The "Space Shuttle" part contains the group of objects that make up the SRMS and its environment. Figure 4 illustrates the group of parts contained within the "Space Shuttle" object. The "Shuttle Electrical Power" part provides power to the "Power Control Panel" part which is used for controlling power to the SRMS. When power is applied to the SRMS, the RMSCP, RMA, RMS Computer, AFDX Routers, and Shuttle Bay systems are energized. The RMS Control Panel provides the operator interface for the whole system. Input commands are processed by the RMS Computer and routed through the AFDX routers to RMA or Shuttle Bay destinations as directed. Subsystem status, sensor, and video data are returned through the AFDX buses to the RMSC for control acknowledgement or information display.

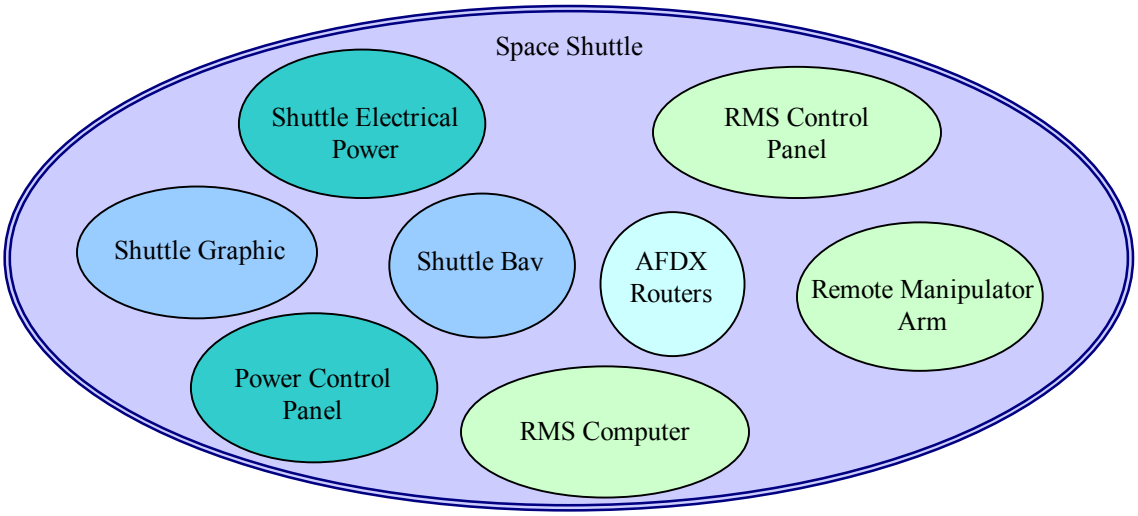


Figure 4: Objects Contained Within Space Shuttle

3.3.2 Part Hierarchy

Because of the relatively large number of parts that typically comprise a simulator, it is usually more convenient to present simulator part hierarchies in table format. Table 1 lists the part hierarchy for the SRMS simulator. The number in the "Level" column indicates the hierarchy level of the parts listed in the "Instance" column of the table.

Table 1: SRMS Simulator Part Hierarchy

Class	Level	Instance
World	0	The World
System	1	System
Earth	2	Earth
SpaceShuttle	2	Space Shuttle
RMSControlPanel	3	RMS Ctrl Panel
VideoMonitor	4	Video 1
VideoMonitor	4	Video 2
ControlPanelGraph	4	CP Graph
RMSComputer	3	RMS Computer
PowerControlPanel	3	Power Ctrl Panel
ShuttleElectPwr	3	Shuttle Elect Pwr
ShuttleBay	3	Shuttle Bay
Camera	4	BayCamFore
CamMotor	5	Pitch Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
CamMotor	5	Yaw Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
CamMotor	5	Zoom Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
MotorController	5	Pitch Controller
MotorController	5	Yaw Controller
MotorController	5	Zoom Controller
Lamp	5	Lamp
ImageSensor	5	ImageSensor
SpaceShuttle3DCamera	6	Camera
AFDX_Router	5	Router A
AFDX_Router	5	Router B
Node	5	Node A

Class	Level	Instance
Node	5	Node B
Camera	4	BayCamAft
CamMotor	5	Pitch Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
CamMotor	5	Yaw Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
CamMotor	5	Zoom Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
MotorController	5	Pitch Controller
MotorController	5	Yaw Controller
MotorController	5	Zoom Controller
Lamp	5	Lamp
ImageSensor	5	ImageSensor
SpaceShuttle3DCamera	6	Camera
AFDX_Router	5	Router A
AFDX_Router	5	Router B
Node	5	Node A
Node	5	Node B
AFDX_Router	4	Router A
AFDX_Router	4	Router B
SpaceShuttle3DGraph	3	Shuttle Graphic
RemoteManipulatorArm	3	RMA
Shoulder	4	Shoulder
ShoulderPitchMotor	5	Pitch Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
ShoulderYawMotor	5	Yaw Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder

Class	Level	Instance
ShaftEncoder	6	High Spd Encoder
MotorController	5	Pitch Controller
MotorController	5	Yaw Controller
AFDX_Router	5	Router A
AFDX_Router	5	Router B
Node	5	Node A
Node	5	Node B
UpperArm	4	Upper Arm
Camera	5	Camera
CamMotor	6	Pitch Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
CamMotor	6	Yaw Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
CamMotor	6	Zoom Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
MotorController	6	Pitch Controller
MotorController	6	Yaw Controller
MotorController	6	Zoom Controller
Lamp	6	Lamp
ImageSensor	6	ImageSensor
SpaceShuttle3DCamera	7	Camera
AFDX_Router	6	Router A
AFDX_Router	6	Router B
Node	6	Node A
Node	6	Node B
AFDX_Router	5	Router_A
AFDX_Router	5	Router_B
StrainGage	5	Strain Gage
DataModule	5	Data Module
Node	5	Node A
Node	5	Node B
Elbow	4	Elbow
ElbowMotor	5	Pitch Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic

Class	Level	Instance
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
MotorController	5	Pitch Controller
AFDX_Router	5	Router A
AFDX_Router	5	Router B
Node	5	Node A
Node	5	Node B
LowerArm	4	Lower Arm
Camera	5	Camera
CamMotor	6	Pitch Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
CamMotor	6	Yaw Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
CamMotor	6	Zoom Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
MotorController	6	Pitch Controller
MotorController	6	Yaw Controller
MotorController	6	Zoom Controller
Lamp	6	Lamp
ImageSensor	6	ImageSensor
SpaceShuttle3DCamera	7	Camera
AFDX_Router	6	Router A
AFDX_Router	6	Router B
Node	6	Node A
Node	6	Node B
AFDX_Router	5	Router_A
AFDX_Router	5	Router_B
StrainGage	5	Strain Gage
DataModule	5	Data Module
Node	5	Node A
Node	5	Node B
Wrist	4	Wrist
WristPitchMotor	5	Pitch Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope

Class	Level	Instance
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
WristYawMotor	5	Yaw Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
WristRollMotor	5	Roll Motor
DCMotor	6	Motor
OscScope	7	Motor Current Scope
MotorGraph	7	Motor Graphic
PlanetaryGearSet	6	Gear
ShaftEncoder	6	Low Spd Encoder
ShaftEncoder	6	High Spd Encoder
MotorController	5	Pitch Controller
MotorController	5	Yaw Controller
MotorController	5	Roll Controller
AFDX_Router	5	Router A
AFDX_Router	5	Router B
Camera	5	Camera
CamMotor	6	Pitch Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
CamMotor	6	Yaw Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
CamMotor	6	Zoom Motor
DCMotor	7	Motor
OscScope	8	Motor Current Scope
MotorGraph	8	Motor Graphic
PlanetaryGearSet	7	Gear
ShaftEncoder	7	Low Spd Encoder
ShaftEncoder	7	High Spd Encoder
MotorController	6	Pitch Controller
MotorController	6	Yaw Controller
MotorController	6	Zoom Controller
Lamp	6	Lamp
ImageSensor	6	ImageSensor
SpaceShuttle3DCamera	7	Camera

Class	Level	Instance
AFDX_Router	6	Router A
AFDX_Router	6	Router B
Node	6	Node A
Node	6	Node B
Node	5	Node A
Node	5	Node B
EndEffector	4	Effector Arm
AFDX_Router	5	Router_A
AFDX_Router	5	Router_B
Ground	5	Ground
Node	3	Node A
Node	3	Node B
AFDX_Router	3	Router A
AFDX_Router	3	Router B
Satellite	2	Satellite
Background	1	Background

3.3.3 Part Descriptions

The subset of part classes within the part tree hierarchy that interface directly with the RMS Computer are described in this section. Each part is identified in a separate sub-section with a description comprising the following:

- A high level description of the part
- All sub-parts that make up the part
- A block diagram of the interconnection of the sub-parts
- The functional requirements of the part
- Any special features or fault operation of the part

In the figures supplied for each part, the yellow boxes represent subparts. The top label on the box is the part (instance) name and the next label is the part class. Signals flow into the part are on the left side of the box, and out of the part on the right except for parts at the lowest level where only the pins are identified. Below is a partial summary list of the unique part classes used in the SRMS simulator.

Descriptions for the parts listed below have been included to aid in understanding the RMS Computer ES implementation described herein. For a description of other significant parts used in the SRMS simulator, please refer to the Ancillary Simulator Parts Document (SARP-I583-204).

Camera	Lamp	RMSComputer
DataModule	MotorController	RMSControlPanel
ImageSensor	RemoteManipulatorArm	SpaceShuttle

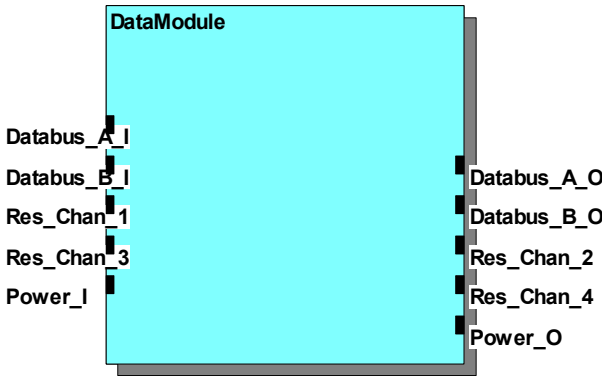
3.3.3.1 Camera

Class Name	**Camera**	
Sub-Part Class	**Sub-Part Instance**	
CamMotor	Pitch Motor Yaw Motor Zoom Motor	
MotorController	Pitch Controller Yaw Controller Zoom Controller	
Lamp	Lamp	
Image Sensor	Image Sensor	
AFDX_Router	Router A Router B	
Node	Node A Node B	
Signal Input	**Signal Output**	**Signal Type**
Databus_A_I	Databus_A_O	Sig AFDX
Databus_B_I	Databus_B_O	Sig AFDX
Power_I	Power_O	Sig Thev
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	<Container Part>	
Limitations	None	


Camera
Integer #1
Integer #2
Integer #3
Integer #4
Integer #5
Databus_A_I
Databus_B_I
Power_A_I
Power_B_I
Pitch Motor
CamMotor
DrivePos_I
DrivePos_O
DriveNeg_I
DriveNeg_O
LSSynchroOut
HSSynchroOut
Shaft_I
Shaft_O
Case_I
Case_O
Pitch Controller
MotorController
Integer #1
DrivePos_I
DrivePos_O
DriveNeg_I
DriveNeg_O
LSSynchroIn
HSSynchroIn
Databus_A_O
Databus_B_O
Power_A_O
Power_B_O
Databus_A_I
Databus_B_I
Power_A_I
Power_B_I
Zoom Controller
MotorController
Router A
AFDX_Router
Databus_1_I
Databus_1_O
Databus_2_I
Databus_2_O
Databus_3_I
Databus_3_O
Databus_4_I
Databus_4_O
Databus_5_I
Databus_5_O
Databus_6_I
Databus_6_O
Power
Node B
Node
6
CONN_1_I
CONN_1_O
CONN_2_I
CONN_2_O
CONN_3_I
CONN_3_O
CONN_4_I
CONN_4_O
CONN_5_I
CONN_5_O
CONN_6_I
CONN_6_O
Databus_A_O
Databus_B_O
Power_A_O
Power_B_O
Yaw Motor
CamMotor
Yaw Controller
MotorController
Lamp
Lamp
Router B
AFDX_Router
Zoom Motor
CamMotor
ImageSensor
ImageSensor
Node A
Node

3.3.3.2 DataModule

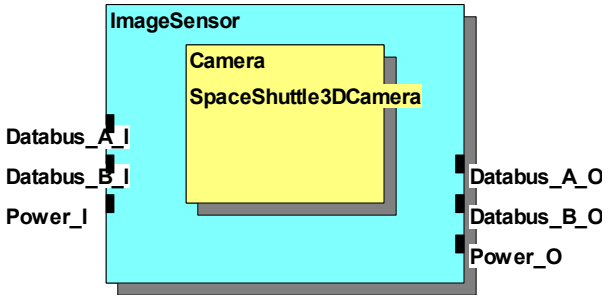
Class Name	DataModule	
Sub-Part Class	**Sub-Part Instance**	
None		
Signal Input	**Signal Output**	**Signal Type**
Databus_A_I	Databus_A_O	Sig AFDX
Databus_B_I	Databus_B_O	Sig AFDX
Res_Chan_1		Sig Voltage
Res_Chan_2		Sig Voltage
Res_Chan_3		Sig Voltage
Res_Chan_4		Sig Voltage
Power		Sig Thev
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	AFDXSig afdxout double chanVoltage[4]	
Requirements	1. Handle AFDX Commands QueryStatusAll QueryResponse, return Strain Gage resistance 2. Receive Strain Gage signal	
Limitations	None	



3.3.3.3 ImageSensor

Class Name	**ImageSensor**	
Sub-Part Class	**Sub-Part Instance**	
SpaceShuttle3DCamera	Camera	
Signal Input	**Signal Output**	**Signal Type**
Databus_A_I	Databus_A_O	Sig AFDX
Databus_B_I	Databus_B_O	Sig AFDX
Power		Sig Thev
Address/Data Bus	N/A	

Direct Function Call	void ImageSensor::SetCameraPosition(double x, double y, double z) void ImageSensor::SetCameraAngle(double yaw, double pitch, double zoom)
Auto Test Function Call	None
User Interface Input	None
User Interface Monitor	None
Internal State Variables	AFDXSig afdxout; unsigned char camerabm[320 * 240 * 3 + 1];
Requirements	Send Image Data in response to AFDX commands
Limitations	None



3.3.3.4 Lamp

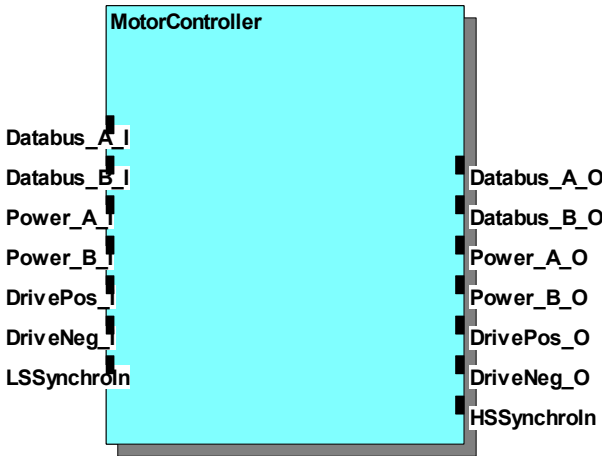
Class Name	**Lamp**	
Sub-Part Class	**Sub-Part Instance**	
None		
Signal Input	**Signal Output**	**Signal Type**
Databus_A_I	Databus_A_O	Sig AFDX
Databus_B_I	Databus_B_O	Sig AFDX
Power		Sig Thev
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	Tbd	
Requirements	Control lamp brightness based on AFDX commands	
Limitations	None	

3.3.3.5 MotorController

Class Name	**MotorController**	
Sub-Part Class	**Sub-Part Instance**	
None		
Signal Input	**Signal Output**	**Signal Type**
Databus_A_I	Databus_A_O	Sig AFDX
Databus_B_I	Databus_B_O	Sig AFDX
DrivePos		Sig Thev
DriveNeg		Sig Thev
LSSynchroIn		Sig Synchro
HSSynchroIn		Sig Synchro
Power_A		Sig Thev
Power_B		Sig Thev
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	

Internal State Variables	AFDXSig afdxout;
	
	double velocity;
	
	enum {
	PCM_EVENT,
	POLL_EVENT,
	};
	
	AddrDataGroup *adgrptr;
	BOOL power_on;
	BOOL opt1_state;
	BOOL opt2_state;
	int optstate;
	int newoptstate;
	int cw;
	int ccw;
	
	short pcm_percent;
	short pcm_us_period;
	short ton;
	short toff;
	BOOL pos_direction;
	BOOL onState;
	BoolSig bfalse;
	BoolSig btrue;
	
	short measRpm;
	short rpmErr;
	short measAccel;
	short lastMeasRpm;	

	short vMax; short aMax; short setPointRpm; short setPointAccel; short pcmPercent; short lastVelSet; short Kp; short Ki; short Kd; short dcErr; short pTerm; short sumErr; short iTerm; short dTerm; double motorShaftAngle; double motorShaftAngleLast; double lowSpeedAngle;
Requirements	Use a PID algorithm to control DC motor speed. Output pulse code modulated voltages to DC motor. Accept AFDX velocity commands.
Limitations	None



3.3.3.6 *RemoteManipulatorArm*

Class Name	**RemoteManipulatorArm**
Sub-Part Class	**Sub-Part Instance**
Shoulder	Shoulder
UpperArm	UpperArm
Elbow	Elbow
LowerArm	LowerArm
Wrist	Wrist
EffectorArm	EndEffector

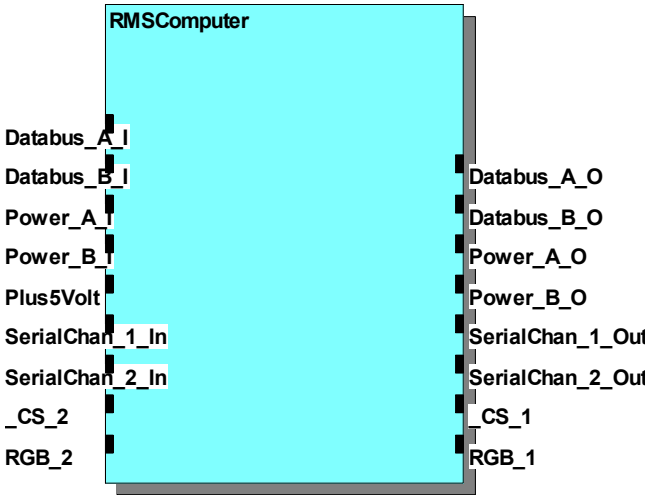
Signal Input	Signal Output	Signal Type
End1_Databus_A_I	End1_Databus_A_O	Sig AFDX
End1_Databus_B_I	End1_Databus_B_O	Sig AFDX
End2_Databus_A_I	End2_Databus_A_O	Sig AFDX
End2_Databus_B_I	End2_Databus_B_O	Sig AFDX
End1_Force_I	End1_Force_O	Sig Force
End2_Force_I	End2_Force_O	Sig Force
End1_Power_A		Sig Thev
End1_Power_B		Sig Thev
End2_Power_A		Sig Thev
End2_Power_B		Sig Thev
Ground		Sig Double

Address/Data Bus	N/A
Direct Function Call	None
Auto Test Function Call	None
User Interface Input	None
User Interface Monitor	None
Internal State Variables	None
Requirements	<Container Part>
Limitations	None

3.3.3.7 RMSComputer

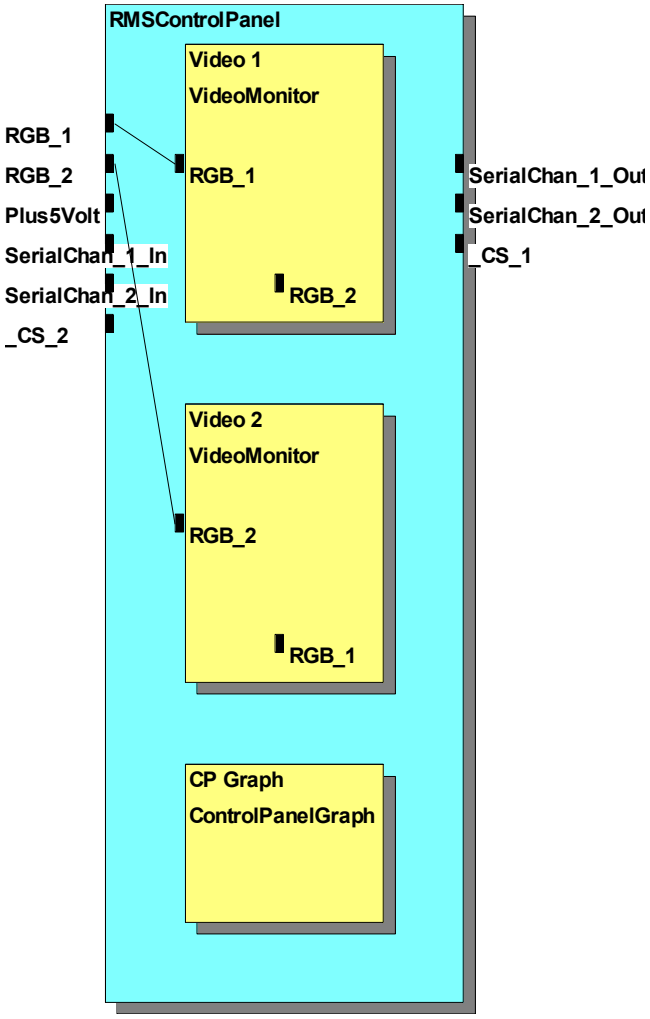
Class Name	**RMSComputer**	
Sub-Part Class	**Sub-Part Instance**	
None		
Signal Input	**Signal Output**	**Signal Type**
Databus_A_I	Databus_A_O	Sig AFDX
Databus_B_I	Databus_B_O	Sig AFDX
Power_A_In		Sig Thev
Power_B_In		Sig Thev
	Plus5Volt	Sig Voltage
SerialChannel_1_In	SerialChannel_1_Out	Sig SPI
SerialChannel_2_In	SerialChannel_2_Out	Sig SPI
	RGB_1	Sig RGB
	RGB_2	Sig RGB
_CS_1		Sig Bool
_CS_2		Sig Bool
Address/Data Bus	N/A	
Direct Function Call	None	

Auto Test Function Call	None
User Interface Input	None
User Interface Monitor	None
Internal State Variables	None
Requirements	Controls SRMS operation
Limitations	None



3.3.3.8 RMSControlPanel

Class Name	**RMSControlPanel**	
Sub-Part Class	**Sub-Part Instance**	
VideoMonitor	Video 1 Video 2	
ControlPanelGraph	CP Graph	
Tbs	Switches, Controls, Indicators	
Signal Input	**Signal Output**	**Signal Type**
Plus5Volt		Sig Voltage
SerialChannel_1_In	SerialChannel_1_Out	Sig SPI
SerialChannel_2_In	SerialChannel_2_Out	Sig SPI
RGB_1		Sig RGB
RGB_2		Sig RGB
	_CS_1	Sig Bool
	_CS_2	Sig Bool
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	Manual control & display of RMS	
Limitations	None	


RMSControlPanel
Video 1
VideoMonitor
RGB_1
RGB_2
RGB_1
RGB_2
Plus5Volt
SerialChan_1_In
SerialChan_2_In
_CS_2
SerialChan_1_Out
SerialChan_2_Out
_CS_1
Video 2
VideoMonitor
RGB_2
RGB_1
CP Graph
ControlPanelGraph

3.3.3.9 SpaceShuttle

Class Name	**SpaceShuttle**	
Sub-Part Class	**Sub-Part Instance**	
AFDX_Router	Router A Router B	
Node	Node A Node B	
ShuttleElectPwr	Shuttle Elect Pwr	
PowerControlPanel	Power Ctrl Panel	
RMSControlPanel	RMS Ctrl Panel	
RemoteManipulatorArm	RMA	
RMSComputer	RMS Computer	
ShuttleBay	Shuttle Bay	
SpaceShuttle3DGraph	Shuttle Graphic	
Signal Input	**Signal Output**	**Signal Type**
RMA_Databus_A_I	RMA_Databus_A_O	Sig AFDX
RMA_Databus_B_I	RMA_Databus_B_O	Sig AFDX
RMA_In_Force_I	RMA_In_Force_O	Sig Force
RMA_Out_Force_I	RMA_Out_Force_O	Sig Force
RMA_Power_A		Sig Thev
RMA_Power_B		Sig Thev
RMA_GROUND		Sig Thev
G_1		Sig Gravity
G_2		Sig Gravity
Address/Data Bus	N/A	
Direct Function Call	None	
Auto Test Function Call	None	
User Interface Input	None	
User Interface Monitor	None	
Internal State Variables	None	
Requirements	<Container Part>	
Limitations	None	


SpaceShuttle
RMS Ctrl Panel
RMSControlPanel
Power Ctrl Panel
PowerControlPanel
RMA
RemoteManipulatorArm
Node B
Node
Router B
AFDX_Router
RMS Computer
RMSComputer
Shuttle Elect Pwr
ShuttleElectPwr
Shuttle Bay
ShuttleBay
Node A
Node
Router A
AFDX_Router
Shuttle Graphic
SpaceShuttle3DGraph
RMA_Databus_A_I
RMA_Databus_B_I
RMA_Power_A_I
RMA_Power_B_I
RMA_GROUND_I
RMA_Out_Force_I
RMA_In_Force_I
G_1_I
G_2_I
RMA_Databus_A_O
RMA_Databus_B_O
RMA_In_Force_O
RMA_Out_Force_O
RMA_Power_A_O
RMA_Power_B_O
RMA_GROUND_O
G_1_O
G_2_O

3.4 RMS Computer ES Implementation

A detailed description of the SRMS design may be found in the System Design Document (SARP-I583-101). Since the RMS Computer ES part is the only ES within the simulator that will be the target for substitution with a DE, this section is devoted the functioning of the RMS Computer ES. Figure 8 shows the system block diagram as given in the SyDD.

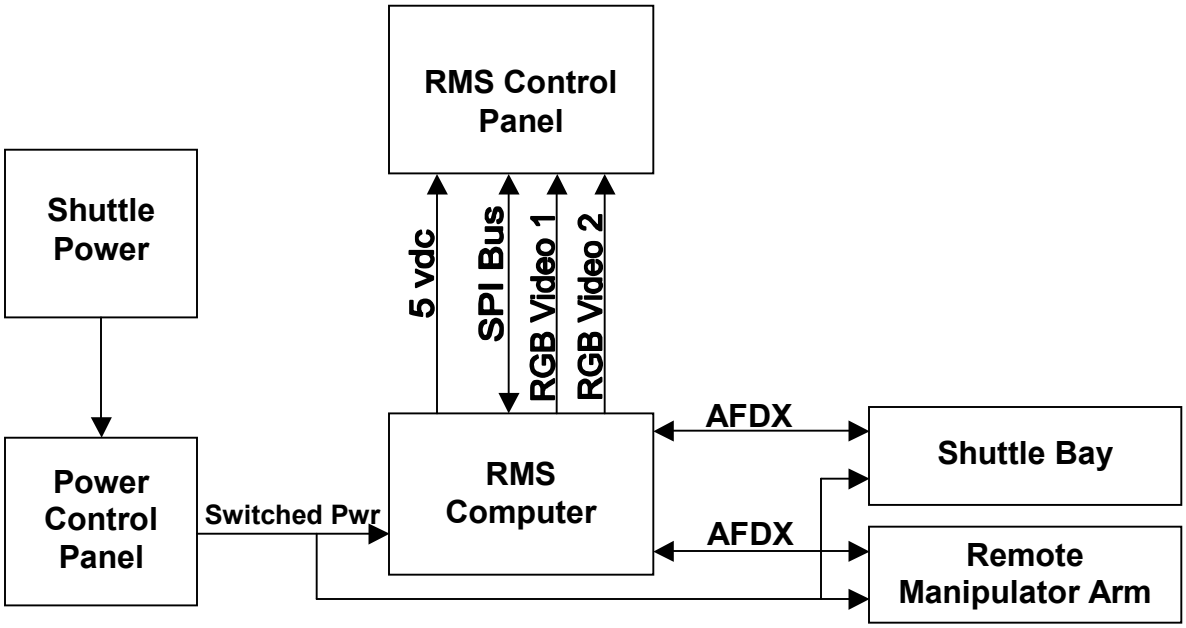


Figure 5: Shuttle RMS Block Diagram

The RMS Computer controls the entire system in response to commands entered via the RMS Control Panel, and data received from the RMA and shuttle bay subsystems. The simulated industry standard Serial Peripheral Interface (SPI) bus provides the data link between the RMS Computer and the RMS Control Panel. Video data for the control panel video monitors is supplied via simulated RGB signals. Communication between the RMS Computer and all other subsystems is accomplished through simulated aviation industry standard Avionics Full-Duplex Ethernet (AFDX) buses

Message formats used for communication between the computer and all of the various SRMS subsystems are specified in the System Design Document (SARP-I583-101). Table 2 lists the RMS Computer ES code written to specify the boundary connections with other simulator parts.

Table 2: RMS Computer ES Part Connection Code

```
RMSComputer::RMSComputer(SimArgs *pargs, Sim *container, char *name): Sim(container, name)
{
  char buff[256];
  sprintf(buff, "Creating Class RMSComputer, part name %s", name);
  Message(buff);

  SaveClassName("RMSComputer");
  int n;
```

```
  // Copy pargs to args. Used for auto .sim file generation
  for(n = 1; n <= pargs->NumAddrDataGroups(); n++)
    args.AddAddrDataGroupPtr(pargs->GetAddrDataGroupPtr(n));
  if(pargs->GetAmbigAddrSpecPtr())
    args.AddAmbigAddrSpecPtr(pargs->GetAmbigAddrSpecPtr());
  for(n = 1; n <= pargs->NumStrings(); n++)
    args.AddString(pargs->GetString(n));
  for(n = 1; n <= pargs->NumInts(); n++)
    args.AddInt(pargs->GetInt(n));
  for(n = 1; n <= pargs->NumDoubles(); n++)
    args.AddDouble(pargs->GetDouble(n));

// 3.0 Hardware Interface

// Refer to Section 3.6 of the Hardware Design Document (SARP-I583-201) for a description
// of the interface design requirements for the RMS Computer. Redundant interface
// signals are not a requirement and have not been implemented, however,a duplicate
// set of interface signals have been created and connected as an example.

  // Register Signals
  RegSigID(Databus_A_I, "Databus_A_I", "Sig AFDX");
  RegSigID(Databus_A_O, "Databus_A_O", "Sig AFDX");
  RegSigID(Databus_B_I, "Databus_B_I", "Sig AFDX");
  RegSigID(Databus_B_O, "Databus_B_O", "Sig AFDX");
  RegSigID(Power_A_I, "Power_A_I", "Sig Thev");
  RegSigID(Power_A_O, "Power_A_O", "Sig Thev");
  RegSigID(Power_B_I, "Power_B_I", "Sig Thev");
  RegSigID(Power_B_O, "Power_B_O", "Sig Thev");
  RegSigID(Plus5Volt, "Plus5Volt", "Sig Voltage");
  RegSigID(SerialChan_1_Out, "SerialChan_1_Out", "Sig SPI");
  RegSigID(SerialChan_1_In, "SerialChan_1_In", "Sig SPI");
  RegSigID(SerialChan_2_Out, "SerialChan_2_Out", "Sig SPI");
  RegSigID(SerialChan_2_In, "SerialChan_2_In", "Sig SPI");
  RegSigID(SerialChan_Data_1_Out, "SerialChan_Data_1_Out", "Sig SPI");
  RegSigID(SerialChan_Data_1_In, "SerialChan_Data_1_In", "Sig SPI");
  RegSigID(SerialChan_Data_2_Out, "SerialChan_Data_2_Out", "Sig SPI");
  RegSigID(SerialChan_Data_2_In, "SerialChan_Data_2_In", "Sig SPI");
  RegSigID(_CS_1, "_CS_1", "Sig Bool");
  RegSigID(_CS_2, "_CS_2", "Sig Bool");
  RegSigID(_CS_DATA_1, "_CS_DATA_1", "Sig Bool");
  RegSigID(_CS_DATA_2, "_CS_DATA_2", "Sig Bool");
  RegSigID(RGB_1, "RGB_1", "Sig RGB");
  RegSigID(RGB_2, "RGB_2", "Sig RGB");

  SimArgs cargs;
  AmbigAddrSpec aas;
  AmbigAddr aa;
  AddrRange ar;

  sprintf(buff, "Making signal connections in class RMSComputer");
  Message(buff);
}
```

3.4.1 RMS Control Panel Communications

The following sub-sections show excerpts of RMS Computer ES code that was written to communicate with the control panel. For the complete RMS Computer code listing refer to the ES source file (RMSComputer.cpp).

3.4.1.1 Receiving SPI Bus Data

Section 3.5.2 of the System Design Document (SARP-I583-101) describes the format of incoming message from the RMS Control Panel. Table 3 shows an example of the ES code used to receive data over the simulated SPI Bus.

Table 3: RMS Computer ES SPI Bus Receive Data Code

```
void RMSComputer::HandleSignal(int id, Sig *data)
{
  AFDXSig afdxsigin;
  AFDXSig afdxsigout;
  SPISig spisig;
  BoolSig bsig;
  short joint, cam, axis, ga;

  switch(id) {

// 3.3 RMS Control Panel Communication
  // Detailed descriptions of the SPI bus message formats are documented in
  // section 3.2 of the SWDD (SARP-I583-102).

// 3.3.1 RMS Control Panel Input Requirements
  // All communications to & from the control panel shall use the industry
  // standard Serial Peripheral Interface (SPI) data bus.  Data reception
  // shall commence following the high to low transition of the
  // chip select (_CS_1) signal.  Incoming SPI data words shall be 16 bits
  // in length and cumulatively stored on each successive clock pulse.
  // SPI bus packet size shall not exceed 16 words.  The end of an SPI
  // packet shall be indicated by a low to high transition of the
  // chip select signal.

    // Handle SPI input bus Chip Select Signal
    case _CS_1:
      bsig = *((BoolSig *)data);

      // look for chip select high to low transition to start reading command
      if(_cs_1 && !bsig.tf) {
        wordcount = 0; // Prepare for new Control Panel data packet
        _cs_1 = FALSE;
      }
      // look for chip select low to high transition to execute command
      // _CS_1 going high signals end of control panel data reception
      else if(!_cs_1 && bsig.tf) {
        ExecuteCommand(); // Process new input from Control Panel
        _cs_1 = TRUE;
      }
      break; // End _CS_1
```

```
    // Handle Incoming Serial SPI data from Control Panel
    case SerialChan_1_In:

      // Accumulate maximum of 16 command words
      // (MAX_SPI_WORDS defined in 'RMSComputer.h')
      // all SPI words are 16 bits long
      if(wordcount < MAX_SPI_WORDS)
        spiChan1Data[wordcount++] = (short)((SPISig *)data)->data;

      break; // End SerialChan_1_In
  }
}


void RMSComputer::ExecuteCommand()
{
  unsigned short majorCommandCode;
  short joint;
  short ctrlFn, btn;
  short camCtrlWord;

  RGBSig rgbsig;
  AFDXSig afdxsigin;
  AFDXSig afdxsigout;

  majorCommandCode = spiChan1Data[0]; // Retrieve new message type code
  switch(majorCommandCode) {

// 3.3.1 Mode Selection

    case SelectMode: // Process control mode button inputs
      if(mode != spiChan1Data[1]) { // Potential mode change
        modeChange = TRUE;
        mode = spiChan1Data[1];
      }
      break;

// 3.3.2 RMA joint motor selection inputs

    case SelectMotor: // Process joint select button inputs
      selectedMotor = spiChan1Data[1];
      break;

// 3.3.3 Manual Control Inputs

    case ManualCommand: // Process command button inputs
        cmdChange = TRUE;
        manualCmd = spiChan1Data[1];
      break;

// 3.3.4 Angle Seek Inputs

    case AngleSeek: // Process angle seek selection inputs
      joint = spiChan1Data[1];
```

```
      seekAngle[joint] = spiChan1Data[2];
      seekAngleFract[joint] = spiChan1Data[3];
      newSeekAngle[joint] = TRUE;

      break;
// etc. for all incoming SPI message types
  }
}
```

3.4.1.2 Sending SPI Bus Data

Section 3.5.2 of the System Design Document (SARP-I583-101) describes the format of outgoing messages from the RMS Control Panel. Table 3 shows an example of the ES code used to send data over the simulated SPI Bus.

Table 4: RMS Computer ES SPI Bus Data Transmission Code

```
// 3.3.2 Output Messages to RMS Control Panel
  // As with incoming messages, output messages to the RMS Control panel shall be
  // sent over a dedicated SPI output bus.  These shall be used to send data to
  // RMA joint status display, control the backlight of the manual control
  // buttons, and to light or extinguish the 'Fault' lamp.  Data packet
  // transmission shall commence by asserting the chip select (_CS_DATA_1) line,
  // sending the data packet with a size not to exceed 16 words, and deasserting
  // the chip select line.  Data word size shall be 16 bits.  SPI output message
  // formats are defined in the SW Design Requirements (SARP-I583-102) section 3.2.3.

// 3.3.2.1 Manual Command button group back-light control
  // The RMS Computer shall light the appropriate control panel manual
  // command lamp when moving the associated RMA joint.

      // Determine appropriate command button to light
      switch(motorSpeedCmd[joint]) {
      case ZERO:
        manCmd = STOP;
        break; // End ZERO

      case MAX_SPEED_CCW:
        manCmd = DEC;
        break; // End MAX_SPEED_CCW

      case MAX_SPEED_CW:
        manCmd = INC;
        break; // End MAX_SPEED_CW
      }
      bsig.tf = FALSE;
      SendSignal(_CS_DATA_1, &bsig);
      spisig.data = ManualCmdData;
      SendSignal(SerialChan_Data_1_Out, &spisig);
      spisig.data = joint;
      SendSignal(SerialChan_Data_1_Out, &spisig);
      spisig.data = manCmd; // Target manual command button
      SendSignal(SerialChan_Data_1_Out, &spisig);
      bsig.tf = TRUE;
      SendSignal(_CS_DATA_1, &bsig);
```

3.4.1.3 Sending RGB Video

Table 5: RMS Computer ES RGB Video Data Transmission Code

```
// 3.3.5 Video display monitor camera source selection

   case Video1Select: // Process new camera source request for display 1
     vidsrc1 = spiChan1Data[1];

     // Point to video buffer for selected camera
     rgbsig.bitmapptr = camBitMap[vidsrc1];

     // Send video from selected camera to upper control panel display
     SendSignal(RGB_1, &rgbsig);
     break;

   case Video2Select: // Process new camera source request for display 2
     vidsrc2 = spiChan1Data[1];

     // Point to video buffer for selected camera
     rgbsig.bitmapptr = camBitMap[vidsrc2];

     // Send video from selected camera to lower control panel display
     SendSignal(RGB_2, &rgbsig);
     break;
```

3.4.2 AFDX Device Communications

The following sub-sections show examples of RMS Computer ES code that was written to communicate with SRMS subsystem parts via the AFDX bus. For the complete RMS Computer code listing refer to the ES source file (RMSComputer.cpp).

3.4.2.1 Sending Data to AFDX Devices

Section 3.5.3 of the System Design Document (SARP-I583-101) describes the format of outgoing AFDX messages from the RMS Control Panel. Table 6 shows an example of the ES code used to send data over the simulated AFDX Bus.

Table 6: RMS Computer ES Code Example for Sending Data to AFDX Devices

```
      // Build & send command to motor
      afdxsigout.msgType = AFDXSig::ExecuteCmd;
      afdxsigout.address = rmaMotorAFDXAddr[joint];
      afdxsigout.data[1] = MotorVelocity; // Data type = motor velocity
      afdxsigout.data[2] = motorSpeedCmd[joint]; // Motor velocity
      SendSignal(Databus_A_O, &afdxsigout); // Issue command
```

3.4.2.2 Receiving Data From AFDX Devices

Section 3.5.3 of the System Design Document (SARP-I583-101) describes the format of incoming AFDX messages from the RMS Control Panel. Table 7 shows an example of the ES code used to receive data via the simulated AFDX Bus.

Table 7: RMS Computer ES code for Receiving Data from AFDX Devices

```
// 3.4 AFDX device communication
  // The RMS computer shall manage all communication with devices
  // connected to the AFDX serial data bus.

// 3.4.1 Data Received from AFDX devices

    case Databus_A_I: // Handle all incoming AFDX data
      afdxsigin = *((AFDXSig *)data);

// 3.4.1.1 AFDX RMA joint motor controllers

      // Handle AFDX inputs from RMA motor controllers
      for(joint = 0; joint < NumRMAJoints; joint++) {
        if(afdxsigin.data[0] == rmaMotorAFDXAddr[joint]) { // Address match?
          switch(afdxsigin.data[1]) { // What type of data is it?

          case Status: // Status data
            rmaJointStatus[joint] = afdxsigin.data[2];
            break; //End Status

          case JointAngle: // Joint angle data
            rmaJointStatus[joint] = afdxsigin.data[2];
            measAngle[joint] = (int)afdxsigin.data[3];
            measAngleFract[joint] = (int)afdxsigin.data[4];
            freshAngleData[joint] = TRUE;
            jointAngleRqstd[joint] = FALSE;
            break; //End JointAngle

          case MotorVelocity: // Motor shaft velocity
             // Data not currently used
            break; // End MotorVelocity

          case MotorAngle: // Motor shaft position
             // Data not currently used
            break; // End MotorAngle
          }
        }
      }
```

4 System testing

Refer to the System Test Design Document for information related to system testing.